Attorney's Docket No. P2534-585

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of	
David Ewing	Group Art Unit: 2173
Application No.: 09/672,957	Examiner: Bautista, Xiomara L.
Filed: September 29, 2000	Appeal No.:
For: METHOD FOR DRAGGING AND DROPPING BETWEEN MULTIPLE LAYERED WINDOWS	

BRIEF FOR APPELLANT

Mail Stop APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This appeal is from the decision of the Primary Examiner dated January 7, 2004, finally rejecting claims 1-26, which are reproduced as an Appendix to this brief.

A check covering the ☐ $165.00 (2402) ☒ $330.00 (1402) Government fee and two extra copies of this brief are being filed herewith.

The Commissioner is hereby authorized to charge any appropriate fees under 37 C.F.R. §§1.16, 1.17, and 1.21 that may be required by this paper, and to credit any overpayment, to Deposit Account No. 02-4800. This paper is submitted in triplicate.

I. Real Party in Interest

The subject application is assigned to Apple Computer. Inc., a California Corporation.

II. Related Appeals and Interferences

There are no other known appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in this appeal.

III. Status of Claims

The subject application contains claims 1-26, all of which are pending and stand finally rejected. This appeal is directed to all of the pending claims.

IV. Status of Amendments

An Amendment is being filed concurrently with this Brief, to correct an informality in some of the dependent claims. The Amendment is reflected in the claims that appear in the Appendix.

V. Summary of the Invention

The claimed invention is directed to graphical user interfaces for computer systems, and more particularly to drag-and-drop operations that can be performed within these types of interfaces. Typically, a user might have multiple windows open on the desktop provided by the user interface. Figure 1 illustrates an example in which two windows, 10 and 14, are open on the desktop. The window which receives user input commands is referred to as the "active" window, and the other windows are inactive. In the example of Figure 1, window 12 is the active window, and window 14 is an inactive window. The active window appears at the topmost layer of the desktop, while the inactive windows appear in lower layers, and therefore can be partially or totally obscured by the active window. In the example of Figure 1, the upper right quadrant of the inactive window 14 is obscured by the active window 12. (Page 1, line 26 to page 2, line 6).

The present invention is directed to the operation in which a user drags an object, such as an icon, to an inactive window. If the desired destination of the object is an obscured portion of the inactive window, the user may be required to execute several steps in order to place the icon at the desired location. One example of the process to be undertaken by the user is depicted in the flow chart of Figure 2, described on page 2, lines 10-28 of the specification. To avoid the need for such a cumbersome procedure, the claimed invention operates to automatically bring the destination window to the forefront of the display, if the user drags the object within the confines of that window and maintains it there for a predetermined period of time.

Figures 5-7 depict a drag-and-drop operation in accordance with the claimed invention. In Figure 5, the user has selected an icon labeled "Folder A" in the active window, Window 1. The user then drags the selected icon from Window 1 to the inactive window, Window 2. As illustrated in Figure 6, if the selected icon remains within Window 2 for a predetermined amount of time, Window 2 becomes the active window and is brought to the forefront of the desktop. Once this occurs, the user can move the selected icon to any desired position within the window, including a previously obscured portion of the window, as illustrated in Figure 7, and drop it at that location. (Page 6, line 24 to page 7, line 14). Once the drop has occurred, the reference address of the icon is changed to Window 2, and all inactive windows are returned to their original layered order (Figure 9B, Step 995; page 10, lines 7-11).

After dragging a selected icon to an inactive window and holding it there sufficiently long for the window to be brought to the forefront, as illustrated in Figure 6, the user may decide not to drop the icon into that window. In such a case, the user can simply drag the icon outside the confines of all of the open windows, as illustrated in Figure 8. When this occurs, Window 1 is returned to its status as the active window, and brought back to the forefront of the display. (Page 7, line 25 to page 8, line 3).

As a further feature of the invention, the functionality described above can also be implemented by pressing a predetermined key on the keyboard. After the use has selected an icon to be dragged, pressing the predetermined key causes the

active window to become the bottom-most window on the display. As a result, the inactive window which was immediately beneath the active window moves to the forefront of the display, so that the user can easily drop an icon into it. (Page 7, lines 15-24; page 9, lines 6-14; Figure 9B, Steps 960 and 965).

VI. The Issues

The final Office Action presents a single issue for a review on this appeal, namely whether claims 1-26 are anticipated by the Gibson reference, European Published Patent Application 0 514 307 A2.

VII. Grouping of Claims

Appellant does not consider all rejected claims to stand or fall together. Rather, even if one of more of the independent claims is found to be anticipated by the Gibson reference, dependent claims recite features of the invention that are separately patentable. The grounds for the separate patentability of various claims are set forth in the following arguments.

VIII. Argument

A. The Gibson reference does not disclose each and every element recited in the claims.

In the final Office Action, all pending claims were rejected as being anticipated by the Gibson reference (EP 0514307). The Gibson reference discloses a graphical user interface that causes viewports, e.g. windows, to be brought to the front of the display as a visual indicator, e.g. cursor pointer, traverses them. Referring to Figure 2, the user desires to drag a note from a mail drawer viewport 22 to a printer icon that is obscured from view. Figure 3 shows that when the user drage the note icon 64 outside the boundary 68 of the viewport 22, a new viewport 20 is brought to the topmost position (column 8, lines 6-10). The user can then drag the note icon 64 onto the printer icon 50.

The reference discloses an override feature in the event that the user does not want a viewport to automatically move to the front of the display. Referring to

Figure 5, if the user presses a dedicated key on the keyboard, when the visual indicator 19 is dragged outside of the boundary 80 of the viewport 72, the viewport 76 does not move to the front. With this override operation in effect, the viewport 76 does not obscure the user's view of destination viewport 78 as the drag operation continues.

The Gibson reference does not anticipate the claimed subject matter since, among other things, it does not disclose the use of a temporal delay in the restacking of overlapping windows. For instance, claim 1 recites the steps of: "starting a timer..." and "displaying said first inactive window...*if* the icon is found to be held within a visible portion of said first inactive window *until* said timer is expired." The Gibson does not disclose such a timer, nor the claimed operation associated with such a timer. In a similar manner, claim 25 recites the step of displaying a second window as an active window "based on whether the icon is held within a visible portion of the second window *for a predetermined amount of time*." Again, the restacking of viewports in the Gibson reference is not conditioned upon the cursor being held within a viewport for a predetermined amount of time. Rather, the restacking occurs automatically in response to detection that the visual indicator 19 crosses the boundary 68 of a viewport. See, for example, column 8, lines 6-10 and column 9, lines 2-6.

In response to Appellant's previous arguments along these lines, the final Office Action refers to the override function disclosed in the Gibson reference, and alleges that the disclosure of this function "suggests a timer." This position is unsupportable, for at least two reasons.

First, it is to be noted that the claims have been rejected as being anticipated, under 35 U.S.C. §102. As set forth in MPEP §2131, "a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference", citing *Verdegaalbros. V. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). As acknowledged in the Office Action, there is no express teaching of a timer in the Gibson reference. Nor is the use of a timer inherent to the override operation. Rather, the override is dependent *only* upon whether the user depresses a dedicated

key, as described at column 8, lines 36-51. The non-temporal nature of this operation is further exemplified by steps 105-111 in the flow chart of Figure 7. As can be seen therein, once the visual indicator is being moved, a determination is made at step 107 whether the indicator has left the source viewport. If so, a further determination is made at step 109 whether the override key is depressed. If not, the viewports are rearranged to bring the target viewport to the top position. If the override key is depressed, no such rearrangement occurs. There is no measurement of time, nor deliberate use of delays, in this operation.

Accordingly, the Gibson reference fails to anticipate the claims, for at least the reason that it neither expressly, nor inherently, discloses the use of a timer or a predetermined delay, in determining whether to restack the viewports.

B. The Gibson reference teaches away from the claimed subject matter

Second, the disclosure of the override function in the Gibson reference teaches *away* from using a timer in the manner of the present invention. The purpose of the override function is to prevent the automatic restacking of the viewports under certain conditions. For example, with reference to Figures 5 and 6, the Gibson reference describes a situation in which the user drags an icon from a source viewport 72 to a destination viewport 78. However, during the dragging operation, the icon crosses the viewport 76. If the viewports are automatically restacked, in accordance with the teachings of the Gibson reference, this would cause the intermediate viewport 76 to be brought to the foreground of the display, and totally obscure the destination viewport 78, as depicted in Figure 6. To prevent this unintended result, therefore, the Gibson patent discloses the ability to override the automatic restacking function, by depressing a designated key while the icon is being dragged across the viewport 76.

When a delay is utilized, as in the present invention, this override feature is unnecessary. Referring again to the example in Figures 5 and 6 of the Gibson reference, in the present invention the viewport 76 would be brought to the foreground only if the icon being dragged remains within this viewport for a predetermined period of time, e.g. until the timer expires. If the user drags the cursor

from the viewport 72 to the viewport 78 without hovering over the viewport 76 for such a period of time, the viewport 76 is not brought to the foreground. Hence, it can be seen that the use of a predetermined delay before bringing a window to the foreground dispenses with the need for an override function of the type disclosed in the Gibson reference.

Accordingly, the disclosure of the override feature in the Gibson evidences a lack of appreciation of the use of a timer as in the claimed invention.

The Office Action appears to focus upon the statement in the Gibson reference that the override feature allows the user to "temporarily" prevent the automatic restacking of overlapping viewports, with reference to column 3, lines 30-35. However, the temporary nature of the override feature is not conditioned upon the use of a timer. Rather, this statement refers to the fact that the override persists only for as long as the user depresses the dedicated key. As stated at column 9, lines 28-31, "The dedicated key on keyboard 14 is *maintained* in a depressed condition *until* visual indicator 19 is relocated within the boundaries of target viewport 78" (emphasis added). In other words, the override is temporary because it only functions while the user is depressing the key. There is no timer associated with the temporary nature of this operation.

C. Claims 4, 10, 16 and 22

These claims are directed to the further feature of the invention depicted in Figure 8 of the application, wherein dragging the selected icon outside of the open windows causes the windows to return to the original display order. For example, claim 4 recites the step of "returning all open windows to the original display layered order, if the icon is monitored to be outside of all open windows."

In rejecting these claims, the final Office Action refers to the Gibson reference at column 11, lines 21-58. This portion of the reference describes a feature which automatically returns the viewports to the arrangement that existed prior the automatic rearrangement that occurs when a visual indicator is moved between viewports. The patent does *not* disclose, however, that the return to the original arrangement is caused by moving a selected icon outside of all open windows, as

recited in the claims. Rather, the reference only discloses that the viewports are returned to their original positions as a result of the user depositing an icon in a target viewport. This feature is described in the reference at column 10, lines 5-33, with reference to the flow chart of Figure 7. As can be seen therein, once the viewports have been rearranged to elevate a target viewport to the top position, at step 11, the user deposits an icon in the target viewport at step 113. In response to this depositing action, a determination is then made at step 115 whether the viewports were rearranged. If so, they are returned to their original position at step 117.

It can be seen, therefore, that the action of returning the viewports to their original positions occurs in response to the user depositing an icon in a target viewport. The reference is silent as to what happens when the user drags an icon outside of all open viewports. It may be the case that the windows remain in their current configuration, rather than being returned to the original arrangement. For this additional reason, therefore, the Gibson reference does not anticipate the subject matter of claims 4, 10, 16 and 22.

IX. Conclusion

In summary, the Gibson reference does not disclose, nor otherwise suggest, interposing a predetermined delay before bringing an inactive window to the forefront of a display, in response to the positioning of a selected icon within that window. As such, it does not disclose each and every element of the rejected claims.

The rejection of claims is not properly founded in the statute, and should be reversed.

Respectfully submitted,

Burns, Doane, Swecker & Mathis, L.L.P.

Date August 5, 2004

By: 

James A. LaBarre
Registration No. 28,632

P.O. Box 1404
Alexandria, Virginia 22313-1404
(703) 836-6620

APPENDIX A

The Appealed Claims

1. A method for manipulating a plurality of windows on a display, comprising the steps of:

displaying a plurality of cascaded, open windows on a display to establish an original display layered order, wherein an active window is the window on a first display layer, windows on a display layer other than the first display layer are inactive windows and at least one of said inactive window is partially hidden;

receiving an indication of an icon being selected;

receiving an indication of the icon being dragged;

monitoring the current location of the icon;

starting a timer, if the icon is found being within a visible portion of first one of said inactive windows; and

displaying said first inactive window on the first display layer, if the icon is found to be held within a visible portion of said first inactive window until said timer is expired.

2. A method of claim 1, further comprising the steps of:

receiving an indication of the icon being released onto said active window;

changing reference memory address of the icon respectively; and

returning all inactive windows to the original display layered order respectively.

3. The method of claim 2, further comprising the steps of:

receiving an indication of a predetermined function key being pressed; and

sending the window on the first display layer to the bottom-most layer.

4. The method of claim 2, further comprising the steps of:

returning all open windows to the original display layered order, if the icon is monitored to be outside of all open windows.

5. The method of claim 1, further comprising the steps of:

receiving an indication of a predetermined function key being pressed; and

sending the window on the first display layer to the bottom-most layer.

6. The method of claim 1, further comprising the steps of:

returning all open windows to the original display layered order, if the icon is monitored to be outside of all open windows or upon receiving an indication of the icon being unselected.

7. A computer readable medium containing a program which executes the steps of:

receiving an indication of an icon being selected;

receiving an indication of the icon being dragged monitoring the current location of the icon; starting a timer, if the icon is found being within a visible portion of first one of said inactive windows; and

displaying said first inactive window on the first display layer, if the icon is found to be held within a visible portion of said first inactive window until said timer is expired.

8. The computer readable medium of claim 7, further executes the steps of:

receiving an indication of the icon being released onto said active window;

changing reference memory address of the icon respectively; and

returning all inactive windows to the original display layered order respectively.

9. The computer readable medium of claim 8, further executes the steps of:

receiving an indication of a predetermined function key being pressed; and

sending the window on the first display layer to the bottom-most layer.

10. The computer readable medium of claim 8, further executes the step of:

returning all open windows to the original display layered order, if the icon is monitored to be outside of all open windows.

11. The computer readable medium of claim 7, further executes the steps of:

receiving an indication of a predetermined function key being pressed; and

sending the window on the first display layer to the bottom-most layer.

12. The computer readable medium of claim 7, further executes the step of:

returning all open windows to the original display layered order, if the icon is monitored to be outside of all open windows or upon receiving an indication of the icon being unselected.

13. A computer system comprising:

means for displaying a plurality of cascaded, open windows on a display to establish an original display layered order, wherein an active window is the window on a first display layer, windows on a display layer other than the first display layer are inactive windows and at least one of said inactive window is partially hidden;

means for receiving an indication of an icon being selected;

means for receiving an indication of the icon being dragged;

means for monitoring the current location of the icon;

means for starting a timer, if the icon is found being within a visible portion of first one of said inactive windows; and

means for displaying said first inactive window on the first display layer, if the icon is found to be held within a visible portion of said first inactive window until said timer is expired.

14. The computer of claim 13, further comprising:

means for receiving an indication of the icon being released onto said active window;

means for changing reference memory address of the icon respectively; and

means for returning all inactive windows to the original display layered order respectively.

15. The computer of claim 14, further comprising:

means for receiving an indication of a predetermined function key being pressed; and

means for sending the window on the first display layer to the bottom-most layer.

16. The computer of claim 14, further comprising:

means for returning all open windows to the original display layered order, if the icon is monitored to be outside of all open windows.

17. The computer of claim 13, further comprising:

means for receiving an indication of a predetermined function key being pressed; and

means for sending the window on the first display layer to the bottom-most layer.

18. The computer of claim 13, further comprising:

means for returning all open windows to the original display layered order, if the icon is monitored to be outside of all open windows or upon receiving an indication of the icon being unselected.

19. A computer system comprising:

a display device for displaying a plurality of cascaded, open windows on a display to establish an original display layered order, wherein an active window is the window on a first display layer, windows on a display layer other than the first display layer are inactive windows and at least one of said inactive window is partially hidden;

a cursor control device for receiving a request for selecting an icon and for moving the selected icon; and

a processor for monitoring the current location of the icon, wherein the processor starts a timer if the icon is found being within a visible portion of first one of said inactive window and reveals said first inactive window on the first display layer, if the icon is found to be held within a visible portion of said first inactive window until said timer is expired.

20. The computer of claim 19, wherein said cursor control device further receives an indication of the icon being released onto said active window and

in response, the processor changes reference memory address of the icon respectively and returns all inactive windows to the original display layered order respectively.

21. The computer of claim 20, further comprises

a key input device for receiving an indication of a predetermined function key being pressed; and

in response, the processor sends the window on the first display layer to the bottom-most layer.

22. The computer of claim 20, wherein said cursor control device further receives an indication which results in the icon being outside of all open windows; and

in response, the processor returns all open windows to the original display layered order.

23. The computer of claim 19, further comprises

a key input device for receiving an indication of a predetermined function key being pressed; and

in response, the processor sends the window on the first display layer to the bottom-most layer.

24. The computer of claim 19, wherein said cursor control device further receives an indication which results in the icon being outside of all open windows or the cursor control device further receives an indication which results in the icon being unselected; and

in response, the processor returns all open windows to the original display layered order.

25. A method for manipulating a first and second window on a display, comprising the steps of:

receiving an indication of a selected icon being dragged from the first window to the second window;

displaying said second window as an active window based on whether the icon is held within a visible portion of the second window for a predetermined amount of time.

26. A method of claim 25, wherein the first window is an active window and the second window is an inactive window when the indication is received.